PortSIP IVR SDK User Manual for Visual C++

Compatibles with Windows 2000/XP/2003/Vista/7/2008.

http://www.portsip.com

v8.1, Dec 26, 2012

portsip
VoIP Solutions Company

NOTICE

© 2012 PortSIP Solutions, Inc. All intellectual property rights in this publication are owned by PortSIP Solutions, Inc. and are protected by United States copyright laws, other applicable copyright laws, and international treaty provisions. PortSIP Solutions, Inc. retains all rights not expressly granted.

This publication is PortSIP confidential. No part of this publication may be reproduced in any form whatsoever or used to make any derivative work without prior written approval by PortSIP Solutions, Inc.

PortSIP Solutions, Inc. reserves the right to revise this publication and make changes without obligation to notify any person of such revisions or changes. PortSIP Solutions, Inc. may make improvements or changes in the product(s) and/or the program(s) described in this documentation at any time.

Unless otherwise indicated, PortSIP registered trademarks are registered in the United States and other territories. All registered trademarks recognized.

For further information contact PortSIP Solutions, Inc.

Table of Contents

NOTICE 2
Overview, Q & A 4
The IVR SDK 5
Callback events 5
Example 6
Audio codecs define 7
Callback Event Mechanism 8
IVR SDK Initialize and register functions 9
IVR SDK audio codec functions 15
IVR SDK DTMF functions 17
IVR SDK calls functions 18
IVR SDK play wave file to remote side function 21
IVR SDK play backgroud wave file to remote side function 24
IVR SDK Receive DTMF function 25
IVR SDK Caller/Callee infomation function 27
IVR SDK Audio recording functions 28

Overview, Q & A

This document describes the methods and callback events supported by PortSIP IVR SDK libraries.

All samples of the PortSIP IVR SDK are available at
http://www.portsip.com/downloads.html

The Sample source code for Visual C++ available at
http://www.portsip.com/downloads/ivr/VCIVRSample.zip

IMPORTANT:

P1 / P2 mismatch build issues with MSVC 2005

Question

I'm currently trying to build PortSIP Sample with MSVC 2005. It builds fine until the linking process starts and reports: "fatal error C1900: II mismatch between 'P1' version '20060201' and 'P2' version '20050411'". How can I resolve this issue?

[Answer]

The PortSIP SDK files and sample project has built by Visual Studio 2005 which has been installed the Microsoft® Visual Studio® 2005 Service Pack 1 and Visual Studio 2005 Service Pack 1 Update for Windows Vista

So if you using the Visual Studio 2005 without these two SP1, then you will encountering this issue, please download and install these two SP1 and rebuilt the sample project.

The IVR SDK

Callback events

```
typedef enum
{
    SIP_UNKNOWN = 0,
    SIP_REGISTER_SUCCESS = 1, // Register to SIP server succeeded
    SIP_REGISTER_FAILURE,     // Register to SIP server failed

    SIP_INVITE_INCOMING,      //  The call is incoming
    SIP_INVITE_TRYING,            //  The call is trying
    SIP_INVITE_RINGING,           //  The call is ringing
    SIP_INVITE_ANSWERED,      //  The callee has answered this call
    SIP_INVITE_FAILURE,           //  The call is failed
    SIP_INVITE_CLOSED,            //  The call is closed
    SIP_INVITE_UPDATED,           //  The remote party updated this call
    SIP_INVITE_UAS_CONNECTED, //  When the UAS receivd the ACK of 200 OK
    SIP_INVITE_UAC_CONNECTED, //  When the UAC sent the ACK for 200 OK
    SIP_INVITE_BEGINING_FORWARD,//  Whe the SDK begin forwarding the call

    SIP_REMOTE_HOLD,          //  The remote party has hold this call
    SIP_REMOTE_UNHOLD,            //  The remote party has take off the hold

    SIP_TRANSFER_TRYING,      //  The call transfer is trying
    SIP_TRANSFER_RINGING,     //  The transfer call is ringing
    SIP_PASV_TRANSFER_SUCCESS,    //  The passive transfer all is succeeded
    SIP_ACTV_TRANSFER_SUCCESS,    //  The active transfer call is succeeded
    SIP_PASV_TRANSFER_FAILURE,    //  The passive transfer call is failed
    SIP_ACTV_TRANSFER_FAILURE,    //  The active transfer call is failed

    SIP_RECV_PAGERMESSAGE,            //  Received a pager message
    SIP_RECV_BINARY_PAGERMESSAGE, //  Received a binary pager message
    SIP_SEND_PAGERMESSAGE_FAILURE,    //  Send the pager message failed
    SIP_SEND_PAGERMESSAGE_SUCCESS,    //  Send the pager message succeeded

    SIP_ARRIVED_SIGNALING,            //  This event will be fired when the SDK received every SIP message



    SIP_WAITING_VOICEMESSAGE,     //  If have waiting voice message, this event will be fired(MWI)
    SIP_WAITING_FAXMESSAGE,           //  If have waiting fax message, this event will be fired(MWI)

    SIP_RECV_DTMFTONE,                //  This event will be fired when received a DTMF tone
```

```
    SIP_PRESENCE_RECV_SUBSCRIBE,  //  The remote side request subscribe presence state
    SIP_PRESENCE_ONLINE,          //  The contact is go online
    SIP_PRESENCE_OFFLINE,         //  The contact is go offline

    SIP_RECV_OPTIONS,                 //  Received an OPTIONS message out of dialog
    SIP_RECV_INFO,                    //  Received a INFO message in dialog
    SIP_RECV_MESSAGE,                 //  Received a MESSAGE message in dialog
    SIP_RECV_BINARY_MESSAGE,      //  Received a binary MESSAGE in dialog

}SIP_EVENT;
```

Example

For example, A called B and the call was established(original call, between A and B), the A use the **PortIVR_refer** function transfer the B to C:

1: B received this transfer, B calling C(refer call, between B and C)

2: A and B will got the **SIP_TRANSFER_TRYING** event.

3: When refer call the is ringing, A and B will got the **SIP_TRANSFER_RINGING** event
4: If refer call has failed(for example, the C rejected the call), A will got **SIP_ACTV_TRANSFER_FAILURE** event, B will got **SIP_PASV_TRANSFER_FAILURE** event.

5: If C answered the refer call, A will got **SIP_ACTV_TRANSFER_SUCCESS** event and hung up the conversation(original call, between A and B) automatically;

6: B will got the **SIP_PASV_TRANSFER_SUCCESS** event.

7: Transfer call successful, and there have the call between B and C only.

The PASV means passive.
The ACTV means active.

Audio codecs define

```
  // Audio codec
typedef enum
{
  AUDIOCODEC_G729       = 18,    //8KHZ
  AUDIOCODEC_PCMA       = 8,     //8KHZ
  AUDIOCODEC_PCMU       = 0,     //8KHZ
  AUDIOCODEC_GSM        = 3,     //8KHZ
  AUDIOCODEC_G722       = 9,     //16KHZ
  AUDIOCODEC_ILBC       = 97,    //8KHZ
  AUDIOCODEC_AMR        = 98,    //8KHZ
  AUDIOCODEC_AMRWB      = 99,    //16KHZ
  AUDIOCODEC_SPEEX      = 100,   //8KHZ
  AUDIOCODEC_SPEEXWB    = 102,   //16KHZ
  AUDIOCODEC_ISACWB   = 103,  //16KHZ
  AUDIOCODEC_ISACSWB  = 104,  //32KHZ
  AUDIOCODEC_G7221      = 121,   //16KHZ
  AUDIOCODEC_DTMF       = 101
}AUDIOCODEC_TYPE;
```

Audio recording format

```
typedef enum
{
  FILEFORMAT_WAVE = 1,
  FILEFORMAT_OGG,
  FILEFORMAT_MP3
}AUDIO_RECORDING_FILEFORMAT;
```

```
typedef enum
{
    RECORD_RECV = 1,
    RECORD_SEND,
    RECORD_BOTH
}RECORD_MODE;
```

Callback Event Mechanism

IVR Core SDK using an abstract struct(SIPCallbackEvent) to implement callback event report. The abstract struct SIPCallbackEvent define as (In SIPEvent.hxx file):

```
struct SIPCallbackEvent
{
  virtual void onCommand(SIPCallbackCommand * command) = 0;
};
```

All of the callback events was reports by **onCommand()** member function of the abstract struct **SIPCallbackEvent**, and the callback events information transfered by parameter **command** of **onCommand()**.

The **SIPCallbackCommand** class was define in the SIPEvent.hxx file .

Users should derive and define onCommand() function to process callback events. such as:

```
struct MySIPEvent : public SIPCallbackEvent
{
  virtual void onCommand(SIPCallbackCommand * command)
  {
      if (!command)
      {
          return;
      }
  switch(command->getEventType())
  {
  case SIP_REGISTER_SUCCESS:
      m_Registered = true;
      break;
  case SIP_REGISTER_FAILURE:
      break;
  ......
  case SIP_INVITE_INCOMING:
      onSIPInviteIncoming(command);
      break;
  }
  PortIVR_delSIPCallbackCommand (command);
};
MySIPEvent * event = new MySIPEvent;
HANDLE SIPCore = PortIVR_initialize (event, ......);
```

IVR SDK Initialize and register functions

```
void * PortIVR_initialize(SIPCallbackEvent * callbackEvent,
                          TRANSPORT_TYPE transportType,
                          PORTSIP_LOG_LEVEL appLogLevel,
                          const char * logFilePath,
                          int maximumLines,
                          const char * agent,
                          const char * STUNServer,
                          int STUNServerPort,
                          int * errorCode);
```

To initialize the IVR core SDK.

Parameters

callbackEvent	-	The callback event pointer.
transportType	-	Transport of SIP, tt can be set as TRANSPORT_UDP, TRANSPORT_TLS, TRANSPORT_TCP.
appLogLevel	-	Use to set the application log level, the levels defined in the PortSIPTypes.hxx file, if enable the log, then will generate `PORTSIP_APP_LOGFILE_xxxx` log file.
LogFilePath	-	The log file path, the path(folder) MUST is exists.
maximumLines	-	This parameter allows maximum 100 lines.
Agent	-	The User-Agent header to insert in messages.
STUNServer	-	Stun server, it's optional and can be pass NULL if you are not use STUN server.
STUNServerPort	-	Stun server port, it will be ignored if the STUNServer is empty.
errorCode	-	If the function succeeds, this parameter value is 0. If the function fails, this value is a specific error code can be found in PortSIP_Errors.hxx file.

Return Values

If the function succeeds, the return value is a handle to the IVR Core SDK object. If the function fails, the return value is NULL.

Remarks

If you want to use the TLS, you have to install the root certificate into the "Trusted Root Certification Authorities area." Usually the certificate was issued by your ITSP.

```
int PortIVR_setUserInfo(void * portIVRLib,
                        const char * userName,
                        const char * displayName,
                        const char * authName,
                        const char * password,
                        const char * localIP,
                        int localSIPPort,
                        const char * userDomain,
                        const char * SIPServer,
                        int SIPServerPort,
                        const char * outboundServer,
                        int outboundServerPort);
```

Set the user information into SDK.

Parameters

username - Account(User name) of the SIP, usually provided by an IP-Telephony service provider.

displayName - *The display name of user name, you can set it as your like, likes: "James Kend". It's optional.*

authName - Authorization user name (usually equals the username).

Password - Password.

localIP - The local computer IP address to bind (for example: 192.168.1.108), it will be using for send and receive SIP message and RTP packet. If pass this IP as the IPv6 format then the SDK will using IPv6.

localSIPPort - The SIP message transport listener port(for example: 5060).

userDomain - User domain; this parameter is optional and can be pass NULL if you are not use domain.

SIPServer - SIP proxy server IP or domain(for example: xx.xxx.xx.x or sip.xxx.com).

SIPServer Port - Port of the SIP proxy server, (for example: 5060).

outboundServer - Outbound proxy server (for example: sip.domain.com), it's optional and can be pass NULL if you are not use outbound server.

outboundServerPort - Outbound proxy server port, it will be ignored if the outboundServer is empty.

Return Values

If the function succeeds, the return value is 0. If the function fails, the return value is a specific error code can be found in PortSIP_Errors.hxx file.

```
void PortIVR_unInitialize(void * portIVRLib);
```

To un-initialize the IVR Core SDK and release resources.

Parameters

PortIVRLib - Handle to the IVR Core SDK object. The PortIVR_initialize function returns this handle.

```
int PortIVR_getLocalIP(void * portIVRLib, int index, char * IP, int ipLength);
```

To get local computer IP address

Parameters

index - The IP address index, for example, the PC has two NICs, we want to obtain the second NIC IP, then set this parameter 1. The first NIC IP index is 0.

IP - A pointer to the buffer that receives the IP

IPLength - The IP parameter buffer size, don't let it less than 16.

Return Value

If the function succeeds, the return value is 0, otherwise the return value is a specific error code can be found in PortSIP_Errors.hxx file.

```
int PortIVR_getLocalIP6(void * portIVRLib, int index, char * IP, int ipLength);
```

To get local computer IP address in IPv6 format.

Parameters

index - The IP address index, for example, the PC has two NICss, we want to obtain the second NIC IP, then set this parameter 1. The first NIC IP index is 0.

IP - A pointer to the buffer that receives the IP

IPLength - The IP parameter buffer size, don't let it less than 64.

Return Value

If the function succeeds, the return value is 0, otherwise the return value is a specific error code can be found in PortSIP_Errors.hxx file.

```
int PortIVR_getNICNums(void * portIVRLib);
```

Obtain the NIC(Network Interface Card) numbers.

Return Value

If the function succeeds, the return value is NIC numbers, otherwise the return value is a specific error code can be found in PortSIP_Errors.hxx file.

```
void PortIVR_setLicenseKey(HANDLE portIVRLib, const char * key);
```

Set the license key into SDK.

Parameters

key - The license key, provide by PortSIP Solutions, Inc. (http://www.portsip.com)

```
int  PortIVR_enableSessionTimer(void * portIVRLib, int timerSeconds,
                                SESSION_REFRESH_MODE refreshMode);
```

This function allows to periodically refresh Session Initiation Protocol (SIP) sessions by sending repeated INVITE requests. The repeated INVITE requests, or re-INVITEs, are sent during an active call leg to allow user agents (UA) or proxies to determine the status of a SIP session. Without this keepalive mechanism, proxies that remember incoming and outgoing requests (stateful proxies) may continue to retain call state needlessly. If a UA fails to send a BYE message at the end of a session or if the BYE message is lost because of network problems, a stateful proxy does not know that the session has ended. The re-INVITES ensure that active sessions stay active and completed sessions are terminated.

Parameters

timerSeconds - The value of the refresh interval in seconds. Minimum requires 90 seconds.

refreshMode - Allows you set the refresher is UAS or UAC.

Return Value

If the function succeeds, the return value is 0, otherwise the return value is a specific error code can be found in PortSIP_Errors.hxx file.

```
void PortIVR_disableSessionTimer(void * portIVRLib);
```

Disable the session timer.

```
int PortSIP_registerServer(void * SIPCoreLib, int expires);
```

Register to SIP proxy server.

Parameters

expires - Registration refresh Interval.

Return Values

If the function succeeds, the return value is 0, otherwise the return value is a specific error code can be found in PortSIP_Errors.hxx file.

```
void PortIVR_unRegisterServer(void * portIVRLib);
```

To un-register from the SIP proxy server.

```
void PortIVR_setSrtpPolicy(void * portIVRLib, SRTP_POLICY srtpPolicy);
```

Set the SRTP policy..

Parameters

srtpPolicy - The SRTP policy is defined as:

If set the policy as **SRTP_POLICY_NONE**, the SDK can receive the encrypted call(SRTP) and

unencrypted call both, but can't place outgoing encrypted call.

If set the policy as **SRTP_POLICY_FORCE**, the SDK just allows receive encrypted Call and place outgoing encrypted call only.

If set the policy as **SRTP_POLICY_PREFER**, the SDK allows receive encrypted and uncrypted call, and allows place outgoing encrypted call and unencrypted call

```
int      PortIVR_setRtpPortRange(void * portIVRLib,
                                 int minimumRtpAudioPort,
                                 int maximumRtpAudioPort,
                                 int minimumRtpVideoPort,
                                 int maximumRtpVideoPort);
```

This function allows set the RTP ports range for audio and video streaming.

Parameters

minimumRtpAudioPort - The minimum RTP port for audio stream.

maximumRtpAudioPort - The maximum RTP port for audio stream.

minimumRtpVideoPort - The minimum RTP port for video stream.

maxmumRtpVideoPort - The maximum RTP port for video stream.

Return Values

If the function succeeds, the return value is 0, otherwise the return value is a specific error code can be found in PortSIP_Errors.hxx file.

Remark

The ports range must is an even number, don't let the audio RTP port range lapped over with video RTP port range, otherwise the function will failed.

```
int PortIVR_setRtcpPortRange(void * portIVRLib,
                             int minimumRtcpAudioPort,
                             int maximumRtcpAudioPort,
                             int minimumRtcpVideoPort,
                             int maximumRtcpVideoPort
                            );
```

This function allows set the RTCP ports range for audio and video streaming.

Parameters

minimumRtcpAudioPort - The minimum RTCP port for audio stream.

maximumRtcpAudioPort - The maximum RTCP port for audio stream.

minimumRtcpVideoPort - The minimum RTCP port for video stream.

maxmumRtcpVideoPort - The maximum RTCP port for video stream.

Return Values

If the function succeeds, the return value is 0, otherwise the return value is a specific error code can be found in PortSIP_Errors.hxx file.

Remark

Don't getting audio RTCP port range repetitious with video RTCP port range, otherwise the function failed.

Don't getting RTCP ports range getting repetitious with RTP port range, otherwise the function failed.

The SDK split port range for each line(session) base on the "maximumLines" parameter which of "initialize" function.

for example, if the port range is 40, the maximum lines are 5, the port range for each line is 40/5 = 8. If the port range for each line is less than 4, the function fails.

Examples:

The "MaximumLines" parameter for "initialize" function is set as 8:

A:

setRtpPortRange(10000, 11000, 12000, 13000); // OK

setRtcpPortRange(10500, 11500, 14000, 15000);// Wrong, the RTCP ports range is repetitious with RTP port range.

setRtcpPortRange(11001, 11100, 14000, 15000);// OK

B:

setRtpPortRange(20001, 20035, 20036, 20089); // OK

setRtcpPortRange(19000, 19019, 14000, 15000);// Wrong, the (port range)/(maximumLines) is less than 4

setRtcpPortRange(19000, 19032, 14000, 15000);// OK

C:

setRtpPortRange(20001, 20035, 20029, 20089); // Wrong, the port range is repetitious

setRtcpPortRange(19000, 19019, 19019, 15000);// Wrong, t the port range is repetitious

```
void PortIVR_delSIPCallbackCommand(SIPCallbackCommand * command);
```

Delete callback command.

Parameters

command - Callback command.

IVR SDK audio codec functions

```
void PortIVR_addAudioCodec(void * portIVRLib, AUDIOCODEC_TYPE codecType);
```

To adding an audio codec.

Parameters

codecType - Audio codec type, defined in the PortSIPTypes.hxx

```
int  PortIVR_setAudioCodecPayloadType(void * portIVRLib, AUDIOCODEC_TYPE codecType,
                                      int payloadType);
```

Set the RTP payload type for dynamic audio codec.

Parameters

codecType - Audio codec type, it allows these values, defined in the PortSIPTypes.hxx

payloadType- RTP payload type.

Return Values

If the function succeeds, the return value is 0, otherwise the return value is a specific error code can be found in PortSIP_Errors.hxx file.

```
int  PortIVR_setAudioCodecParameter(void * portIVRLib, AUDIOCODEC_TYPE codecType,
                                    const char * parameter);
```

This function use to set the audio codec parameter.

Parameters

codecType - The codec type which you want to set the parameter to it.

parameter - The parameter in string format.

Remark

PortIVR_setAudioCodecParameter (AUDIOCODEC_AMR, “mode-set=0; octet-align=1; robust-sorting=0”);

```
void PortIVR_clearAudioCodec(void * portIVRLib);
```

To remove all added audio codecs.

```
bool PortIVR_isAudioCodecEmpty(void * portIVRLib);
```

Detect added audio codecs.

Parameters

PortIVRLib - Handle to the Core SDK object. The PortIVR_initialize function returns this handle.

Return Values

If no audio codec set, the return value is true, otherwise is false..

```
void PortIVR_discardAudio(void * portIVRLib,
                          bool discardIncomingAudio,
                          bool discardOutgoingAudio);
```

Discard the incoming / outgoing audio packets.

Parameters

discardIncomingAudio - If the value is set to true, the SDK will discard all received audio packets.

discardOutgoingAudio - If the value is set to true, the SDK will discard all sending audio packets.

IVR SDK DTMF functions

```
void PortIVR_setDTMFSamples(void * portIVRLib, int samples);
```

Set the samples of DTMF tone.

Parameters

samples - Samples of the DTMF tone, default value is 160

```
int  PortIVR_sendDTMF(void * portIVRLib, long sessionId, char code);
```

Send DTMF tone.

Parameters

sessionId - Session ID of the call.

code - DTMF Digit (1, 2, 3, 4, 5, ... 0, *, #)

Return Value

If the function succeeds, the return value is 0, otherwise the return value is a specific error code can be found in PortSIP_Errors.hxx file.

Remark

For example send the # DTMF tone: ***PortIVR_sendDTMF (mSIPCoreLib, 1, '#');***

IVR SDK calls functions

```
long PortIVR_call(void * portIVRLib, const char *  callTo,
                        bool sendSDP, int * errorCode);
```

Make the call(invite).

Parameters

PortIVRLib - Handle to the IVR Core SDK object. The PortIVR_initialize function returns this handle.

callTo - The callee username(number), it can be "sip:usernumber@sipserver.com" or "usernumber"

sendSDP - To enable/disable send SDP with invite. If set to true, then invite will include SDP, if set to false, then make invite without SDP.

errorCode - If the function succeeds, this parameter value is 0. If the function fails, this value is a specific error code can be found in PortSIP_Errors.hxx file.

Return Value

If the function succeeds, the return value is a session ID to the call, otherwise 0.

```
int      PortIVR_rejectCall(void    *    portIVRLib,    long    sessionId,    int    code,
                            const char * reason);
```

To reject incoming call.

Parameters

PortIVRLib - Handle to the IVR Core SDK object. The PortIVR_initialize function returns this handle.

sessionId - Session ID of the call.

code - Rejected call status code.

reason - Rejected call status text(rejected reason).

Return Values

If the function succeeds, the return value is 0, otherwise the return value is a specific error code can be found in PortSIP_Errors.hxx file.

```
int  PortIVR_answerCall(void * portIVRLib, long sessionId, int waitSeconds);
```

Answer the incoming call.

Parameters

sessionId - Session ID of the call.

waitSeconds - In the SIP, if the callee is answered the call – send a 200 OK message to caller, the caller should reply an ACK message, if the callee received this ACK then means this call is established, otherwise

the callee has to waiting the ACK to established the call.

If set `waitSeconds` as 0, then this function is Non-blocking and does not waiting the ACK after sent 200 OK message, it will returns immediately; If set it more than 0, this function will blocking and returns once received the ACK. If does not received the ACK in waitSeconds, then function will return once timeout.

Return Value

If the function succeeds, the return value is 0, otherwise the return value is a specific error code can be found in PortSIP_Errors.hxx file.

```
int  PortIVR_terminateCall(void * portIVRLib, long sessionId);
```

Terminate the conversation(Hang up the call).

Parameters

sessionId - Session ID of the call.

```
int  PortIVR_hold(void * portIVRLib, long sessionId);
```

To place a call on call.

Parameters

sessionId - Session ID of the call.

Return Value

If the function succeeds, the return value is 0, otherwise the return value is a specific error code can be found in PortSIP_Errors.hxx file.

```
int  PortIVR_unHold(void * portIVRLib, long sessionId);
```

Take off hold.

Parameters

sessionId - Session ID of the call.

Return Value

If the function succeeds, the return value is 0, otherwise the return value is a specific error code can be found in PortSIP_Errors.hxx file.

```
int  PortIVR_refer(void * portIVRLib, long sessionId, const char * referTo,
                        int waitSeconds);
```

Refer the call to another one.

Parameters

PortIVRLib - Handle to the IVR Core SDK object. The PortIVR_initialize function returns this handle.

sessionId - Session ID of the call.

referTo - Target of the refer, it can be "sip:number@sipserver.com" or "number" only

waitSeconds - If set waitSeconds as 0, then this function is Non-blocking and does not waiting, it will returns immediately; If set it more than 0, this function will blocking and returns once Transfer success or Transfer Failure. If does not received the Transfer success or Transfer Failure in waitSeconds, then function will return once timeout.

Return Value

If the function succeeds, the return value is 0, otherwise the return value is a specific error code can be found in PortSIP_Errors.hxx file.

Remark

Example: transfer the call to sip:testuser12@sip.portsip.com:8000

PortIVR_refer(mSIPLib, sessionId, "sip:testuser12@sip.portsip.com:8000");

Or transfer the call to sip:test1@176.11.90.202:

PortIVR_refer(mSIPLib, sessionId, "sip:test1@176.11.90.202");

You can download the demo AVI at: http://www.portsip.com/downloads/blindtransfer.rar, please use the Windows Media Player to play the AVI file after extracted, it will shows how to do the transfer.

```
int  PortIVR_attendedRefer(void * portIVRLib,
                           long sessionId,
                           long replaceSessionId,
                           const char * referTo);
```

To make an attended refer.

Parameters

sessionId - Session ID of the call.

replaceSessionId - Session ID of the replace call.

referTo - Target of the refer, it can be "sip:number@sipserver.com" or "number" only

Return Value

If the function succeeds, the return value is 0, otherwise the return value is a specific error code can be found in PortSIP_Errors.hxx file.

Remark

Please view the sample project to got more details. Or download the demo AVI at: http://www.portsip.com/downloads/video/attendedtransfer.rar please use the Windows Media Player to play the AVI file after extracted, it will shows how to do the attended transfer.

IVR SDK play wave file to remote side function

```
int      PortIVR_startPlayFile(void * portIVRLib,
                               long sessionId,
                               const char * filename);
```

To set a wave file to play to the remote party, the remote side will be hearing this wave file when the call is established.

Parameters

sessionId - Session ID of the call.

szFileName - The name and path of the file to play. must be sign channel 16bit 8000Hz/16000Hz/32000Hz .wav file. likes: c:\\sample.wav

Return Value

If the function succeeds, the return value is 0, otherwise the return value is a specific error code can be found in PortSIP_Errors.hxx file.

```
int  PortIVR_waitPlayFileEvent(void * portIVRLib, long sessionId, int dwMilliseconds);
```

Wait for PortIVR_startPlayFile event.

Parameters

sessionId - Session ID of the call.

dwMilliseconds - Function Maximum waiting time.

Return Value

<0 – the return value is a specific error code can be found in PortSIP_Errors.hxx file.

0 – playfile has completed without interruption

1 – Recv a dtmf event the method

2 – A playstop method has interrupted the method

ECoreIVRWaitingTimeOut - waiting time out

```
int PortIVR_playFile(void * portIVRLib,
                     long sessionId,
                     const char * fileName,
                     const char * stopTones);
```

To set a wave file to play to the remote party, the remote side will be hearing this wave file when the call is established.

Parameters

sessionId - Session ID of the call.

szFileName - The name and path of the file to play. must be sign channel 16bit 8000Hz/16000Hz/32000Hz .wav file. likes:

c:\\sample.wav

szStopTones - Gives a list of tones that will interrupt the play. Allowed values: a string of zero or more digits, including the star (*) and pound sign(#), or a plus sign (+) to indicate all tones..

Return Value

<0 – An error, the return value is a specific error code can be found in PortSIP_Errors.hxx file.

0 – playfile has completed without interruption

1 – A Stoptone has interrupted the method

2 – A playstop method has interrupted the method

Remark

This is synchronous function, it will blocking until file play end.

```
void PortIVR_stopPlayFile(void * portIVRLib, long sessionId);
```

Stop a current play file on a call.

Parameters

sessionId - Session ID of the call.

```
int PortIVR_getPlayFileDetail(void * portIVRLib,
                              long sessionId,
                              char* fileName,
                              int filenNameLength,
                              int* elapseTimeSec,
                              int* totalTimeSec);
```

Get current play file detail.

Parameters

sessionId - Session ID of the call.

szFileName - Return current Playfile name.

filenNameLength - szFileName buffer size

elapseTimeSec - Return the elapsed playing time.

totalTimeSec - Return the file total duration.

```
int PortIVR_seekPlayFile(void * portIVRLib, long sessionId,
                                long offsetSec, int seekOrigin);
```

Seek current play file on a call.

Parameters

sessionId - Session ID of the call.

offsetSec - Number of second to offset from origin.

seekOrigin - osition from where offset is added. It is specified by one of the following constants defined:

0 Beginning of file

1 Current position of the file pointer

2 End of file

IVR SDK play backgroud wave file to remote side function

```
int PortIVR_startPlayFileAsBackground(void * portIVRLib,
                                      long sessionId,
                                      const char * fileName);
```

Play a background wave file to remote, the remote side will be hearing this wave file when the call is established.

Parameters

sessionId - Session ID of the call.

`fileName` - The name and path of the file to play. must be sign channel 16bit 8000Hz/16000Hz/32000Hz .wav file. likes: c:\\sample.wav

Return Value

If the function succeeds, the return value is 0, otherwise the return value is a specific error code can be found in PortSIP_Errors.hxx file.

```
void PortIVR_stopPlayFileAsBackground(void * portIVRLib, long sessionId);
```

Stop a current playfileBackground method

Parameters

PortIVRLib - Handle to the IVR Core SDK object. The PortIVR_initialize function returns this handle.

sessionId - Session ID of the call.

IVR SDK Receive DTMF function

```
int PortIVR_getDigits(void * portIVRLib,
                      long sessionId,
                      int * digitsCount,
                      char * digitsBuff,
                      int digitsBuffLength);
```

Returns any DTMF captured in a buffer.

Parameters

PortIVRLib - Handle to the IVR Core SDK object. The PortIVR_initialize function returns this handle.

sessionId - Session ID of the call.

digitsCount - Return the received DTMF tones count.

digitsBuff - Return the received DTMF tones in string.

digitsBuffLength - Length of DigitsBuff.

Return Value

If the function succeeds, the return value is 0, otherwise the return value is a specific error code can be found in PortSIP_Errors.hxx file.

Remark

This call is Non-blocking. The buffer is not cleared until clearDigits method is called.

```
int PortIVR_waitForDigits(void * portIVRLib,
                          long sessionId,
                          int * digitsCount,
                          char * digitsBuff,
                          const char * stopTones,
                          int maxDigitsCount ,
                          int interDigitDelay,
                          int maximumDelay);
```

Wait For Digits until get enough digits or out time.

Parameters

PortIVRLib - Handle to the IVR Core SDK object. The PortIVR_initialize function returns this handle.

sessionId - Session ID of the call.

digitsCount - Return get digits count.

digitsBuff - Return DTMF captured in this buffer.

maxDigitsCount - Max recvive digits number.

interDigitDelay - Inter digit delay, Maximum time between receive two digits(sec.).

maximumDelay - Maximum digit delay, Maximum wait digits time(sec.).

Return Value

<0 – An error, a specific error code can be found in PortSIP_Errors.hxx file.

0 – A Stoptone has interrupted the method
1 – Max digits count has received
2 – DTMF receive interval over interDigitDelay(sec.)
3 - DTMF receive total time over maximumDelay(sec.)

Remark

This is synchronous function, it will blocking until returns.

The buffer is not cleared before clearDigits method is called.

```
int PortIVR_clearDigits(void * portIVRLib, long sessionId);
```

Remove any DTMF in the buffer.

Parameters

PortIVRLib - Handle to the IVR Core SDK object. The PortIVR_initialize function returns this handle.

sessionId - Session ID of the call.

Return Value

If the function succeeds, the return value is 0, otherwise the return value is a specific error code can be found in PortSIP_Errors.hxx file.

IVR SDK Caller/Callee infomation function

```
int PortIVR_getCaller(void * portIVRLib,
                      long sessionId,
                      char * caller,
                      int callerLength);
```

Get the caller of call.

Parameters

PortIVRLib - Handle to the IVR Core SDK object. The PortIVR_initialize function returns this handle.

sessionId - Session ID of the call.

caller - Return the caller of call.

callerLength - caller buffer length

```
int PortIVR_getCallee(void * portIVRLib,
                      long sessionId,
                      char * callee,
                      int calleeLength);
```

Get the callee of call.

Parameters

PortIVRLib - Handle to the IVR Core SDK object. The PortIVR_initialize function returns this handle.

sessionId - Session ID of the call.

callee - Return the callee of call.

calleeLength - callee buffer length

IVR SDK Audio recording functions

```
int PortIVR_startChannelRecording(void * portIVRLib,
                                  int sessionId,
                                  const char * recordFilePath,
                                  const char * fileName,
                                  bool appendTimestamp,
                                  AUDIO_RECORDING_FILEFORMAT fileFormat,
                                  RECORD_MODE recordMode);
```

Start record the voice.

Parameters

PortIVRLib - Handle to the SIP Core SDK object. The PortSIP_initialize function returns this handle.

sessionId - Session ID of the call.

recordFilePath - File path, for example: c:\\audio

recordFileName - File name, for example: audiofile (without extended name)

appendTimestamp - If set as true then the SDK will append the timestamp to filename automatically.

fileFormat - Record file format, if pass it as **FILEFORMAT_WAVE**, then will be record the audio into a file as WAV format; If pass it as **FILEFORMAT_OGG**, then will be as OGG format; Set to **FILEFORMAT_MP3**, then record as MP3 file.

recordMode - Record file mode,.

RECORD_RECV – Only record receive(speaker) data.

RECORD_SEND - Only record Send(microphone) data.

RECORD_BOTH – Record receive/send data

Return Value

If the function succeeds, the return value is 0, otherwise the return value is a specific error code can be found in PortSIP_Errors.hxx file.

```
int PortIVR_stopChannelRecording(void * portIVRLib, int sessionId);
```

Stop the audio call recording.

Parameters

PortIVRLib - Handle to the SIP Core SDK object.

sessionId - Session ID of the call.

Return Value

If the function succeeds, the return value is 0, otherwise the return value is a specific error code can be found in PortSIP_Errors.hxx file.

```
int PortIVR_getChannelRecordedInfo(void * portIVRLib,
                                   int sessionId,
                                   char * recordedFilename,
                                   int recordedFilenameLength,
                                   int * durationMS);
```

Get recording Filename and duration time.

Parameters

PortIVRLib - Handle to the SIP Core SDK object.

sessionId - Session ID of the call.

recordedFilename - Return recorded filename.

recordedFilenameLength - recordedFilename buffer size.

durationMS - Return record duration time.

Return Value

If the function succeeds, the return value is 0, otherwise the return value is a specific error code can be found in PortSIP_Errors.hxx file.